```
import numpy as np

# Define problem-specific parameters
POP_SIZE = 10      # Number of antibodies (candidate solutions)
CLONE_FACTOR = 3   # Number of clones per selected antibody
MUTATION_RATE = 0.2 # Probability of mutation
ITERATIONS = 50    # Maximum number of iterations

# Define fitness function (Affinity Function)
def fitness_function(x):
    return -np.sum((x - 0.5) ** 2)  # Example: minimizing squared
distance from 0.5

# Initialize population (random real-valued antibodies)
def initialize_population(size, dim=5):
    return np.random.rand(size, dim)

# Select top antibodies based on fitness
def select_best(population, fitness, num_selected):
    indices = np.argsort(fitness)[-num_selected:]  # Select highest
fitness
    return population[indices], fitness[indices]

# Clone selected antibodies
def clone_population(selected, clone_factor):
    clones = np.repeat(selected, clone_factor, axis=0)
    return clones

# Introduce mutations (hypermutation)
def mutate_population(clones, mutation_rate):
    mutations = (np.random.rand(*clones.shape) < mutation_rate) *
np.random.uniform(-0.1, 0.1, clones.shape)
    return np.clip(clones + mutations, 0, 1)  # Ensure values remain
in [0,1]

# Main CLONALG algorithm
def clonal_selection_algorithm():
    population = initialize_population(POP_SIZE)

    for iteration in range(ITERATIONS):
        fitness = np.array([fitness_function(ind) for ind in
population])
        selected, selected_fitness = select_best(population, fitness,
POP_SIZE // 2)
        clones = clone_population(selected, CLONE_FACTOR)
        mutated_clones = mutate_population(clones, MUTATION_RATE)

        # Evaluate fitness of new population (mutated clones +
original population)
        new_population = np.vstack((population, mutated_clones))
```

```
        new_fitness = np.array([fitness_function(ind) for ind in
new_population])

        # Select best individuals for the next generation
        population, _ = select_best(new_population, new_fitness,
POP_SIZE)

        best_fitness = max(new_fitness)
        print(f"Iteration {iteration + 1}: Best Fitness =
{best_fitness:.5f}")

    # Return the best solution found
    best_solution = population[np.argmax([fitness_function(ind) for
ind in population])]
    return best_solution

# Run the Clonal Selection Algorithm
best_solution = clonal_selection_algorithm()
print("Best Solution Found:", best_solution)
```

```
Iteration 1: Best Fitness = -0.34569
Iteration 2: Best Fitness = -0.31528
Iteration 3: Best Fitness = -0.28406
Iteration 4: Best Fitness = -0.27441
Iteration 5: Best Fitness = -0.21922
Iteration 6: Best Fitness = -0.17535
Iteration 7: Best Fitness = -0.17535
Iteration 8: Best Fitness = -0.16212
Iteration 9: Best Fitness = -0.12828
Iteration 10: Best Fitness = -0.12828
Iteration 11: Best Fitness = -0.10455
Iteration 12: Best Fitness = -0.09818
Iteration 13: Best Fitness = -0.07824
Iteration 14: Best Fitness = -0.05976
Iteration 15: Best Fitness = -0.05121
Iteration 16: Best Fitness = -0.04237
Iteration 17: Best Fitness = -0.02936
Iteration 18: Best Fitness = -0.02854
Iteration 19: Best Fitness = -0.02630
Iteration 20: Best Fitness = -0.02620
Iteration 21: Best Fitness = -0.00838
Iteration 22: Best Fitness = -0.00838
Iteration 23: Best Fitness = -0.00830
Iteration 24: Best Fitness = -0.00594
Iteration 25: Best Fitness = -0.00594
Iteration 26: Best Fitness = -0.00423
Iteration 27: Best Fitness = -0.00423
Iteration 28: Best Fitness = -0.00252
Iteration 29: Best Fitness = -0.00174
Iteration 30: Best Fitness = -0.00174
```

```
Iteration 31: Best Fitness = -0.00174
Iteration 32: Best Fitness = -0.00132
Iteration 33: Best Fitness = -0.00029
Iteration 34: Best Fitness = -0.00029
Iteration 35: Best Fitness = -0.00029
Iteration 36: Best Fitness = -0.00029
Iteration 37: Best Fitness = -0.00021
Iteration 38: Best Fitness = -0.00021
Iteration 39: Best Fitness = -0.00021
Iteration 40: Best Fitness = -0.00018
Iteration 41: Best Fitness = -0.00018
Iteration 42: Best Fitness = -0.00018
Iteration 43: Best Fitness = -0.00018
Iteration 44: Best Fitness = -0.00018
Iteration 45: Best Fitness = -0.00018
Iteration 46: Best Fitness = -0.00018
Iteration 47: Best Fitness = -0.00018
Iteration 48: Best Fitness = -0.00007
Iteration 49: Best Fitness = -0.00007
Iteration 50: Best Fitness = -0.00007
Best Solution Found: [0.49492172 0.50302748 0.49841569 0.50568263
0.5022368 ]
```